file being held open and identified as not being defective though containing incorrect data.

This is illustrated in FIG. **13C**, which shows the two possibilities for writing into the EDC/FEI field **334** of a sector. If a sector is to be written with information read from a disk file, and in the reading process of FIG. **13B**, step **384** has been reached (signifying that the data read is corrupted and the forced error indicator code is not present for the sector which was read), then the data known to be corrupted is to be written to a good block of recording medium on the same or another disk surface and the forced error indicator code is to be set when the new sector is written. This is indicated by going or branching from step **384** directly to a step **392**, where the write operation occurs with the forced error indication code set in field **334**. A contrast, when the sector is to be written with new data from a host (which data is presumed to be reliable and uncorrupted) or with data successfully read from storage, as indicated by the "yes" branches from steps **364** or **380**, then the write operation occurs with an appropriate non-complemented error detection code written to the EDC/FEI field. (Step **394**).

The data preamble "space" field **342** is the area necessary to accommodate controller quantization errors in the transition between reading headers and writing data preamble. The length of splice field **344** is the number of words necessary to accommodate worst-case header transmission delays, header compare time, write splice area and PLO lock time. The number for this area (in words) is placed in the "data preamble length" field of the response to the above-designated command.

The length of the write-to-read recovery field **346** is the number of bits necessary for write recovery, plus an allowance for uncertainty.

The length of the reinstruct time field **348** is the disk area traversed during the time the controller is cleaning up the current sector transfer and sending the command to the next one.

The Headers

The sector header is 128 bits: thirty-two bits replicated four times. The layout of one of the thirty-two bit copies is shown in FIG. **12**. A 16-bit word **352** and the lower 12 bits of the next word **354** form a 28-bit block number field, which is followed by a 4-bit header code **356**. The block number field represents an LBN, an RBN, an XBN, or a DBN, depending on the header code. The block number field provides enough addressing for approximately 0.25 giga-sectors or 1 terabit of data.

The octal header code may, for example, be interpreted as follows. First, an exemplary code such as 00 (octal) may indicate a usable logical sector wherein data may or may not be valid, depending upon the validity of the EDC. The block number in the header represents the LBN for this block. This header code may appear in LBN space only. Another code, such as 03, may indicate an unusable revectored logical sector. This header code may appear in the non-RCT portion of LBN space only. The data field contains the RBN header field of the replacement block, replicated 128 times; the block number in the header represents the LBN for this block.

Yet another code, such as 05, may indicate an unusable primary revectored logical sector. Such a sector has been revectored onto the first replacement sector on the track. The data field contains the RBN header field of the replacement block, replicated 128 times. The block number in the header represents the LBN for this block. This LBN has been revectored to its primary RBN. This header field may be registered in the non-RCT portion of LBN space only.

Yet another code, such as 06, may be used to indicate a usable replacement sector, wherein data may or may not be valid, depending on the validity of the EDC. The block number in the header represents the RBN for this block. This header code may appear in RBN space only.

Another code, such as 11, may signify an unusable sector, where data is invalid. The block number in the header is that of the sector's type if it had been a usable sector. This header code may appear in RBN, XBN, or DBN space, the RCT area of LBN space, and in LBN's which have been secondary revectored due to header errors.

Yet another code, such as 12, may signify a usable external sector, wherein data may or may not be valid depending upon the value of the EDC. The block number in the header represents the XBN for this block. This header code may appear in XBN space only. A further code, such as **14**, may represent a usable diagnostic sector. The block number in the header represents the DBN for this block. This header code may appear in DBN space only.

Header Compare Algorithm

A header compare algorithm is used by the controller for locating a designated sector. First, the controller determines the address of the sector it is searching for on the disk (i.e., the "target" address). The controller then reads the four copies of the 32-bit header of the sector that may be at the target address. These headers are broken into two 16 bit fields (low and high). If any two of the four low fields, as retrieved from the disk, match the low field of the target address and any two of the four high fields, as retrieved from the disk, match the high field of the target address, then the header compare succeeds. If at least two low matches are not found, then a header match is not possible.

If at least two low matches are found and two high matches are not found, then it is possible that the correct sector was located but the header code did not match the target header code. This is possible if an LBN has been replaced, or if a bad block has been found in a multi-copy protected area (i.e., RCT, XBN or DBN). The controller alters the header code in the target address then determines if two high matches now exist. A variant of the header compare algorithm is also used to conclude that a drive has mis-seeked or seeked to the wrong cylinder or group, or that an incorrect head has been selected. For this purpose, any three of the four high header words must match and any three of the four low header words must match, since there is not an expected header value to match against. Given this three-way match, the controller may interpret the header code and block number fields to determine the actual cylinder, group and track that have been accessed, for comparison against the correct values.

The Data

The contents of the data field are application-dependent. The data field size will depend on the format used by the host processor. For the assignee's products, there are two basic data field sizes, 512 bytes and 576 bytes. A portion of all disks is always formatted with 512 byte data fields. This is the manufacturing defect area (XBN). The other areas on disk drives attached to those

controllers that support both sector sizes may be formatted in either 512 or 576 byte format. Each time a device comes "online" to a controller, the controller is responsible for determining the sector size employed by the device according to the algorithm described below. First, the device is instructed to change the sector size of its reading operation to 512 bytes. The starting sector of the first copy of format information is read. The first word of this sector is tested. If it is equal to a preselected number, then LBN/RBN space is written in 512 byte mode. On the other hand, if it is written with some other preselected number, then such space is written in 576 byte mode. If the starting XBN of the first copy is not readable or a value other than the aforementioned preselected values is in the first word, then the starting XBN of the next copy of the format control table is computed using the following formula:

next copy XBN=old copy XBN+size of format control table.

This new sector is then read. If it has an uncorrectable I/O error, then the next copy is accessed, until all copies are tried. If all copies are read and there is no copy that can be read without an uncorrectable I/O error, then a media format error is returned to the host. Also, if the first sector (i.e., XBN) of the first copy read without an uncorrectable error contains an invalid media mode code, then a media format error is returned to the host.

(The host may force the device into a specific mode, in which case the controller will attempt to access the device unit using that mode, without issuing the media format error. This is intended only as a means of data recovery, and not as a standard operating practice.)

If the volume is in 512 byte format, the algorithm is complete. If in 576 byte format, the controller is responsible for prefacing all operations on XBN's or 512 byte DBN's with a command to change the size to 512 bytes, and preceeding the next reference to LBN's or RBN's with a command to change the size back to 576 byte format. In other words, the controller is responsible for changing the sector size dynamically based on which space the sector falls in, using 512 byte format for XBN's and DBN's but 576 byte format for LBN's and RBN's.

The EDC

The Error Detecting Code (EDC) is a 16-bit code used to detect errors caused by internal problems in the controller. It is applied as an end-to-end verification of correct controller operation. The algorithm shown here was designed to detect column errors as well as multi-bit parity errors.

The EDC is computed via an exclusive-OR operation and left circular shift algorithm, using a non-zero initial value and 16 bit word size. The rotate used in this algorithm has no carry. The algorithm itself is listed in FIG. 13A. In addition to detecting errors, the EDC also is used herein to provide a forced error indicator. This is accomplished by storing the one's complement of the correct EDC in the EDC field of the sector. An "error" is thereby indicated when the sector is read; this "error" is eliminated when the sector is next written with correct EDC. This technique makes it very easy for diagnostic routines to identify sectors having forced errors. That is, when an EDC indicates an error, it is a simple matter to determine whether that EDC is in fact the one's complement of the EDC to be expected on the basis of the recorded data.

The Track

A track is composed of sectors and timing marks. There must be at least two sectors per track (1 LBN sector and 1 RBN sector). Timing marks are of two types: (1) sector marks and (2) index marks. A sector mark precedes each sector and may be used by the controller for rotational optimization purposes. An index mark precedes the first sector on each track within the first group in the cylinder and precedes a sector at the same angular position with respect to the first group on all other tracks within all other groups in a cylinder.

Detailed Description of an Embodiment of The Logical Layer

Address Spaces

There are four address spaces in the set of sectors made available to the controller by the drive. The first address space contains the set of logical blocks which are visible to the host. This LBN space is divided into two regions: the host accessible area and the RCT's. The second address space contains replacement blocks which are used to replace logical blocks that have become unusable. These RBN's are invisible to the host except for the implications they have on allocation policies. The controller utilizes the logical blocks and replacement blocks in a fashion that presents to the host a logically contiguous set of blocks numbered from zero to H−1, where H is the block capacity as seen from the host. The third address space is the extended block space; (XBN's); this is a set of blocks visible only to the controller, which is used to store manufacturing format control information and transient controller-specific information. Finally, there is the diagnostic block space (DBN's) containing blocks devoted to controller-resident diagnostics. The DBN's are also visible only to the controller. These address spaces are differentiated by unique header codes, preventing inadvertent access to or operation in the wrong type of sector.

Although conformation to the overall geometry described herein is a requirement of the invention, the specific capacities and other physical parameters associated with the geometry of the disk will vary from device type to device type. These specific parameters are part of the permanent characteristics of each device type, and are determined when the device is designed. The controller shields from the host these parameter-dependent device properties. The controller issues a generic command termed the GET CHARACTERISTICS command, in response to which the drive responds by sending to the controller the parameters necessary for use in geometry-related operations. The controller then uses those parameters as appropriate and necessary.

The Drive Characteristics Blocks

As mentioned above, in a secondary storage subsystem according to this invention, a disk drive provides to the controller, responsive to a command, one or more messages containing various parametric information. In this regard, it should be noted that within a drive there may be one or more subunits, each of which can be addressed independently by the host and controller. Thus, to fully characterize the drive, two commands are

used. First, a command named the GET COMMON CHARACTERISTICS command is employed to evoke a message regarding parameters which are common to all subunits of the drive. Next, a comand named the GET SUBUNIT CHARACTERISTICS command is used to evoke the characteristics of specific subunits of a drive. The format of the response to the GET COMMON CHARACTERISTICS command is illustrated in FIG. **14**. There, a 23 byte sequence is shown. The first byte identifies the nature of the response. The lower half of the second byte conveys the length of a short time out, expressed as power of two. The upper half of the second byte contains a number indicating in the version of the bus used between the controller and drive. In the third byte, the drives bit transfer rate is specified, scaled down by a factor of 100,000. The fourth byte, like the second byte, is broken in half. Its lower half includes a long time out, also expressed as a power of two; while its upper half conveys the number of retries of a failed operation which will be required by the drive. In the lower half of the fifth byte, a number is written to indicate the number of FCT and RCT copies maintained. The most significant bit in the fifth byte, SS, indicates the drive sector size. The sixth byte specifies the number of error recovery levels which the drive makes available. It is a characteristic of this system that the controller need not be aware of the error recovery techniques available in the drive. The drive may employ several different error recovery techniques, numbered in their order of increasing or decreasing chance of success. Assume, for example, that by convention error recovery Level 1 corresponds to the technique having the greatest probability of success; error recovery Level 2 is the next most likely to succeed, etc. Then, the controller need only signal for the invocation of error recovery Level 1 and the subsequent error recovery techniques, in ascending numerical order (corresponding to descending probability of success). The drive, responsive to seeing each of the error recovery level indicators, invokes the appropriate recovery method.

The seventh byte contains the ECC threshold, above which replacement and revectoring are invoked. The eighth byte contains an indication of the microcode revision number of the drive and the ninth byte contains an indication of its hardware revision number.

Bytes **10–15** contain a unique drive identification number or serial number. The sixteenth byte contains a drive type identifier and byte seventeen indicates the rotational speed of the disk platters, in revolutions per second.

Bytes **18–23** contain various error thresholds.

The response to the GET SUBUNIT CHARACTERISTICS command is indicated in FIGS. **15A** and **15B**. As shown there, the response is 39 bytes in length. The first byte contains a pattern indicating the nature of the response. Bytes **2**, **3**, **4** and the lower-order half of byte **5** contain the number of cylinders in the LBN space. The field comprising bits **6–4** of byte **5** contains bits number **30–28** of all cylinder numbers on this subunit.

The number of groups per cylinder is indicated in byte **6**.

The lower-order half of byte **7** contains bits **27–24** of the first LBN on this subunit, while the upper-order half of that byte contains the same bits of the first XBN on this subunit. Byte **8** contains the number of tracks per group. Byte **9** is fragmented into two halves, the lower half of which contains the bits **27–24** of the first RBN on this subunit, while the upper half of the byte contains the same bits for the first DBN on this subunit.

The number of RBN's per track is indicated on byte **10**. Bytes **12** and **13** contain the length of the data and header preambles, respectively, in words.

Bytes **14–17** record the media type. Bytes **18** and **19** give the size of copy of the FCT, in XBN's.

Bytes **20–27** are used for the 512 byte format, and their counterpart for the 576 byte format is bytes **28–35**. As labelled in the drawing, the contents of the bytes should be self-explanatory. Bytes **20** and **28** indicate the number of LBN's per track. Bytes **21** and **29** indicate the group offset—i.e., the offset from one group to another to permit spiral read operation. The number of LBN's in the host area is indicated from byte **22** through the lower-half of byte **25** and from byte **30** to the lower-half of byte **33**. Bytes **20–23** and **34–35** indicate the size of a copy of the RCT, in LBN's.

Bytes **36–39** are common to both formats. Bytes **36** and **37** indicate the size of the XBN space, in cylinders. Byte **38** indicates the number of groups in the DBN area and byte **39** indicates the size of the DBN space in cylinders.

The replacement sectors in any given drive are logically numbered from 0 to (Rs−1), where Rs=Lc*g*t*r is the total number of replacement sectors. A replacement block number is converted to a specific physical disk location through a series of transformations performed by the controller using parameters supplied by the drive. These transformations are described later. The last r sectors (where r is a drive-specific parameter) of each track in the host application area is reserved for replacement blocks for revectored bad blocks. These alternate blocks lie outside of the LBN space presented by the controller to the host, and are accommodated in the logical-to-physical address conversion algorithm described below.

FIG. **16** illustrates the first two and last tracks in the LBN/RBN space of a subunit.

External Block Track Geometry

The external sectors on any given drive are logically numbered from (0 to Xtot−1), where Xtot=Xc*g*t*s and is the total number of external sectors.

The transformation for converting an external block number to a specific physical disk location is explained later.

XBN's are allocated contiguously on all XBN cylinders; they increase incrementally from the starting XBN number as the sector number, track number, and cylinder number increase, until the XBN cylinders are exhausted. There are no replacement blocks on XBN cylinders.

FIG. **17** illustrates the first two and last track in the XBN space of a subunit.

Diagnostic Cylinder Geometry

The diagnostic sectors on a drive are numbered logically from 0 to Ds−1, where Ds=Dc*g*t*s is the total number of diagnostic sectors. The method for transforming DBN's to specific physical disk locations is described below. An adequate number of cylinders is reserved for diagnostic usage. Sector headers in those cylinders are coded to reflect that they are DBN's. These diagnostic cylinders are formatted initially in the 512 byte mode and the last cylinder in this space must remain in that mode; that cylinder contains various data

patterns prerecorded at the factory. Diagnostic space geometry is illustrated in FIG. **18**.

Address Conversions

Two generic variables are used to express the address conversion algorithms. They are actual or calculated device characteristics. The function QUO( ) is used to indicate a quotient resulting from a division operation and the function REM( ) is used to indicate the remainder resulting from a division operation.

The starting LBN for a drive (L) is computed from the characteristic "HISTRTLBN", the high order part of the address of the starting LBN. (see below). This is done by OR-ing the "HISTRTLBN" nibble into bits **27–24** of a previously zeroed longword.

Given a header LBN, the algorithm listed in FIG. **19** is used to determine the logical block's physical sector address. In reading that figure, note that the starting cylinder for a drive (C) is computed from the drive characteristic "HI CYL", the high order part of the cylinder address. This is done by OR-ing the "HI CYL" nibble into bits **30–28** of a previously zeroed longword. In the figure, "o" represents an offset.

Given a header RBN, the algorithm of FIG. **20** may be used to determine the replacement block's physical sector address. Note that the starting RBN for a drive (R) is computed from the characteristic "HISTRTRBN," the high order part of the RBN address. This is done by OR-ing the "HISTRTRBN" nibble into bits **27–24** of a previously zeroed longword.

Given a header XBN, the algorithm listed in FIG. **21** may be used to determine the external block's physical sector address. The starting XBN for a drive (X) is computed from the drive characteristic "HISTRTXBN," the high order part of the XBN address. This is done by OR-ing the "HISTRTXBN" nibble into bits **27–24** of a previously zeroed longword.

Given a header DBN, the controller executes the algorithm of FIG. **22** to determine the diagnostic block's physical sector address. The starting DBN for a drive (D) is computed from the characteristic "HISTRTDBN," the high order part of the DBN address. This is done by OR-ing the "HISTRTDBN" nibble into bits **27–24** of a previously zeroed longword.

Given a header LBN that has been revectored to the first RBN on the same track (primary RBN), then the following algorithm or formula may be used to determine the replacement block's RBN:

$$RBN = R + (QUO((LBN - L)/1))*r$$

Given a host LBN that has been revectored to the first RBN on the same track (primary RBN), then the following formula may be used to determine the replacement block's RBN:

$$RBN = R + (QUO((LBN)/1))*r$$

Given the physical address (cylinder, group and track) of a logical block that has revectored to the first RBN on the same track (primary RBN), then the following formula may be used to determine the replacement block's RBN:

$$RBN = R + ([([(Cyl.\ No. - C)*g] + Group\ No.)*t] + Track\ No.)*r$$

Detailed Description of an Embodiment of the Functional Layer Revectoring

Once a sector has been replaced, revectoring should occur upon each access to the replaced LBN. Three revectoring mechanisms are supported by the particular implemention discussed herein. These mechanisms all depend upon values in the code field of the sector's header to initiate revectoring. Additionally, all revectored LBN's contain 128 copies of the replacement block's header in their data field, unless revectoring is the result of a header error. The revectoring mechanisms differ in the ways that the addresses of the target RBN's are determined.

In the primary revectoring mechanism, the position of the RBN to which revectoring is performed is implied by the position of the LBN on the volume. This implied position is the first replacement sector on the track containing the LBN. This is a many LBN to one RBN mapping function.

With so-called secondary revectoring, an arbitrary RBN is used whose address is determined by the presence of the 128 copies of the RBN's header value (code and address) in the data field of the bad LBN. The algorithm listed in FIG. **23** is used to determine reliably the correct value of the RBN header; it provides as output (from the 128 copies input) the address found to have at least 24 matches, if there is one.

Finally, there is a so-called tertiary revectoring mechanism which is used when the header compare algorithm fails to determine a valid header address or code or the algorithm of FIG. **23** fails to yield a valid result. It is important to determine then if the LBN has been revectored or if access to the LBN should result in an unrecoverable error. Since all revectored LBN's are recorded in the multiple copies of the RCT, an RCT search is used to determine if the bad LBN has been revectored. The RCT search algorithm, described above, results in the RBN address if the LBN was revectored, or a failure indication if it was not revectored. The determination that the attempted input/output operation was done to the correct sector requires, since the header is "smashed" and unusable (1) a determination that the correct cylinder, group and track have been selected; (2) for controllers that use sector counting via sector and index pulses, at least one revolution of counting after completion of the foregoing step and (3) for controllers that locate sectors by reading headers, at least four full revolutions searched after the foregoing step is complete. Failure to achieve a header match on the latter two actions requires invocation of tertiary revectoring.

Formatting Support

Formatting and reformatting processes are responsible for establishing which sectors are bad and replacing them, if they are in the host applications area, or formatting there headers with the unusable code if they are bad LBN's in the RCT, bad XBN's, bad DBN's or bad RBN's.

The formatting process is supported by the format control tables (FCT), which are used to record information about the location of manufacturing detected bad blocks. Format information for both 512 byte and 576 byte formats is stored in the FCT. The first subtable in the FCT contains information about where the bad blocks would be located if the disk were located in the 512 byte format, the second subtable contains informa-

tion about where the bad blocks would be located if the disk were recorded in the 576 byte format. For those mass storage devices that don't support the 576 byte format, the 576 byte subtable contains null entries.

A second function of the FCT is the identification of the current mode of the LBN space (i.e., whether it is recorded in 512 or 576 byte format. The first sector of each FCT copy contains a code identifying the current LBN sector size. This mode identification sector is updated each time the volume is formatted.

The FCT contains at least one track of subsystem scratch storage also.

Each copy of the FCT is composed of one volume information block, one 512 byte format table, one 576 byte format table, and one subsystem temporary storage area (distributed amongst the alignment pads). This format is illustrated in FIG. **24**. The XBN area itself is always formatted to contain 512 byte sectors. Sector **0** of the FCT contains various volume identification information. Its format is illustrated in FIG. **25**.

Conclusion

Having thus described an exemplary embodiment of the invention, it will be apparent that various alterations, modifications and improvements will readily occur to those skilled in the art. Such obvious alterations, modifications and improvements, though not expressly described above, are nonetheless intended to be implied and are within the spirit and scope of the invention. Accordingly, the foregoing discussion is intended to be illustrative only, and not limiting; the invention is limited and defined only by the following claims and equivalents thereto.

What is claimed is:

1. In a secondary storage subsystem (**2**) for a data processing system (**10**), wherein data is recorded on a mass storage medium (**5**) and the smallest addressable unit of the medium is a sector (**9**), each sector including a header field for recording address information and a data field for recording data to be associated with and stored at such address, the improvement comprising:

means (**4**, FIG. **13C-392**) for writing in each sector a predetermined code, termed the forced error indicator, when the data being recorded in the data field is known to be logically corrupted and the medium underlying the sector is not known to be defective; and

means (**4**, FIG. **13B-374**) for providing a signal, termed a forced error signal, responsive to detection of the forced error indicator during a read operation.

2. The apparatus of claim **1** wherein the means for writing the forced error indicator code includes

means (**4**, FIG. **13C-394**) for generating for each sector an error detection code which is uniquely related to such sector's data, in accordance with a preselected algorithm, for use in detecting the presence of errors in reading data recorded in the sector; and

means (**3**, **4**, FIG. **13C**) for writing a signal into a predetermined location in the sector (**9**, FIG. **11-334**), such location being termed the EDC/FEI field, said signal being

(a) the error detection code for the data being recorded in the sector when the data is not known to be logically corrupted (**396**), or

(b) the forced error indicator code when the data being recorded in the sector is known to be logically corrupted (**392**).

3. The apparatus of claim **2** wherein the forced error indicator code is the one's complement of the error detection code for the data being recorded in the sector.

4. The apparatus of claim **3** further including:

means (**4**, FIG. **13B-362**) for generating a second error detection code upon reading the data recorded in a sector, using the preselected algorithm; and

the means for providing a forced error signal comprising means for comparing the error detection code thus generated with the signal read from the EDC/FEI field (**4**, FIG. **13B-364**, **370**) and for generating the forced error indicator when the signal read from the EDC/FEI field corresponds to the one's complement of the second error detection code.

5. In a secondary storage subsystem for a data processing system, wherein data is recorded on a mass storage medium and the smallest addressable unit of the medium is a sector, a method of replacing a defective sector with a substitute sector, such that information to be written to or read from a defective sector is written to and then read from the substitute sector instead, once the defective sector is identified as unreliable, such method comprising the steps of:

A. reserving a portion of the medium to be used as spare sectors for replacing defective sectors, at least one spare sector being provided within each set of sectors occupying contiguous logical locations on the medium;

B. replacing the first defective sector in said set of sectors with the first one of said spare sectors, termed a primary replacement sector;

C. indicating such replacement by writing a first predetermined code in the defective sector; and

D. when writing data to or reading data from a sector, detecting the first predetermined code and, in response to detecting said code, revectoring the writing or reading operation to said first one of the spare sectors for the involved set of sectors where the code was detected.

6. The method of claim **5** wherein each sector comprises a header field and a data field, the address of the sector normally being written in the header field and the information to be stored in the sector being written in the data field, and further wherein said predetermined code is written in the header field of the defective sector.

7. The method of claim **6** wherein the reserved sectors are evenly distributed throughout the medium.

8. The method of claim **6** wherein the medium is a magnetic disk and the sets of sectors are tracks.

9. The method of claim **6** wherein the reserved sectors are located in predefined locations within said tracks.

10. The method of claim **6** further including the steps of, when the primary replacement sector is unavailable:

D. selecting for a defective sector other than the first defective sector in said set of sectors a replacement sector other than the primary replacement sector, said replacement sector being termed a secondary replacement sector;

E. in the header field of such defective sector, writing a second predetermined code indicating that said

sector has been replaced by a secondary replacement sector;

F. in the data field of each such defective sector, writing a predetermined multiple number of copies of the physical address of the secondary replacement sector selected therefor;

G. on reading the header field of the defective sector, checking for said second code;

H. responsive to detecting said second code, obtaining the physical address of the secondary replacement sector by reading said multiple copies and comparing them statistically to arrive at the recorded value of said address; and

I. revectoring the writing or reading operation intended for the defective sector to said secondary replacement sector.

11. The method of claim **10** wherein the reserved sectors are evenly distributed throughout the medium.

12. The method of claim **10** wherein the medium is a magnetic disk and the sets of sectors are tracks.

13. The method of claim **10** wherein the primary replacement sectors are located in predefined locations within said tracks.

14. The method of claim **10** further including the steps of, when the primary replacement sector is unavailable:

J. providing on the medium multiple copies of a table containing a list of each spare sector and the address of the defective sector replaced by it, if any;

K. responsive to detecting a defective sector, searching a copy of said table to find the address of the replacement sector therefor; and

E. revectoring the writing or reading operation intended for the defective sector to said replacement sector.

15. In a secondary storage subsystem for a data processing system, wherein data is recorded on a mass storage medium and the smallest addressable unit of the medium is a sector, each sector comprising a header field and a data field, the address of the sector normally being written in the header field and the information to be stored in the sector being written in the data field, a method of replacing a defective sector with a substitute sector, such that information to be written to or read from a defective sector is written to and then read from the substitute sector instead, once the defective sector is identified as unreliable, such method comprising the steps of:

A. reserving a portion of the medium to be used as spare sectors for replacing defective sectors;

B. selecting for a defective sector a replacement sector from among said spare sectors;

C. in the header field of such defective sector, writing a predetermined code indicating that said sector has been replaced by a secondary replacement sector;

D. in the data field of a defective sector, writing a predetermined multiple number of copies of the physical address of the selected replacement sector therefor, termed a secondary replacement sector;

E. on reading the header field of the defective sector, checking for said second code;

F. responsive to detecting said second code, obtaining the physical address of the secondary replacement sector by reading said multiple copies and comparing them statistically to arrive at the recorded value of said address; and

G. revectoring the writing or reading operation intended for the defective sector to said secondary replacement sector.

16. In a secondary storage subsystem for a data processing system, wherein data is recorded on a mass storage medium and the smallest addressable unit of the medium is a sector, a method of replacing a defective sector with a substitute sector, such that information to be written to or read from a defective sector is written to and then read from the substitute sector instead, once the defective sector is identified as unreliable, such method comprising the steps of:

A. reserving a portion of the medium to be used as spare sectors for replacing defective sectors;

B. selecting for a defective sector a replacement sector from among said spare sectors;

C. providing on the medium multiple copies of a table containing a list of each spare sector and the address of the defective sector replaced by it, if any;

D. responsive to detecting a defective sector, searching a copy of said table to find the address of the replacement sector therefor; and

E. revectoring the writing or reading operation intended for the defective sector to said replacement sector.

17. In a disk drive for a secondary storage facility of a data processing system, wherein a read/write head must be positioned to read or write successive portions of the medium and the usable area of the storage medium is divided into sectors, each sector occupying a specific physical position relative to an index location on the medium and being available for reading or writing once per disk rotation, a method of reducing the time consumed in head repositioning, comprising the steps of:

A. logically grouping sectors into tracks, groups and cylinders according to access time latencies, a track being a set of sectors occupying contiguous logical disk locations, a group, being a set of tracks which can be selected within the time required for a sector to rotate past a head, and a cylinder being a collection of groups that can be selected by operations having latencies less than the time for a head-positioning seek operation;

tracks, groups and cylinders being independent of physical organization of the drive; and

B. mapping the physical address of each sector to a logical track, group and cylinder address to effect optimal access time reduction.

18. In a secondary storage device, the improvement comprising: dividing the medium into multiple address spaces, at least two address spaces being addressable by a host computer system which uses the mass storage device and at least two address spaces being invisible to and not accessible by the host computer system;

the first address space addressable by the host computer system being the set of storage locations visible to an operating system of the host computer;

a second address space addressable by the host computer system being a space containing revector control tables for revectoring access to bad blocks on the medium;

the first address space not accessible by the host computer system comprising a region which provides formatting information;

a second address space not accessible by the host computer being adapted to contain diagnostic information.

* * * * *